Lecture 11

CS 170

Sanjam Garg.

Edit Distance

Input : $x[1....n]$ & $y[1....m]$

Goal : Find minimum # of keystrokes needed to edit x to y

1 keystroke is needed to add a character
remove a character
substitute a character

Example #

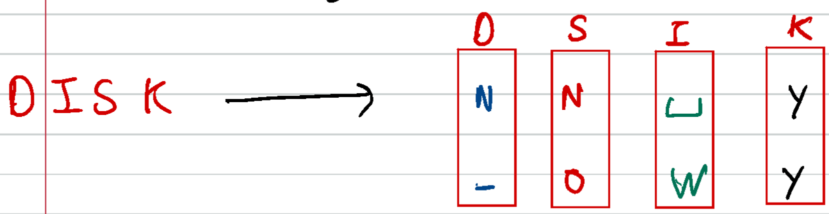
CAP → CUP →1

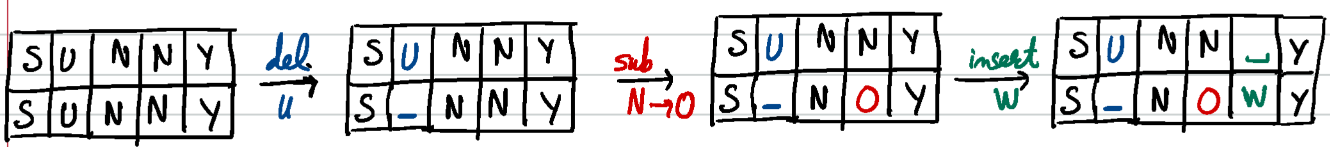
AAPPL → APPLE →2

SUNNY → SNOWY

SUNNY $\xrightarrow[U]{\text{del}}$ SNNY $\xrightarrow[N \to O]{\text{sub}}$ SNO_Y $\xrightarrow[W]{\text{insert}}$ SNOWY

How to visualize?





Step 1: Define subproblems

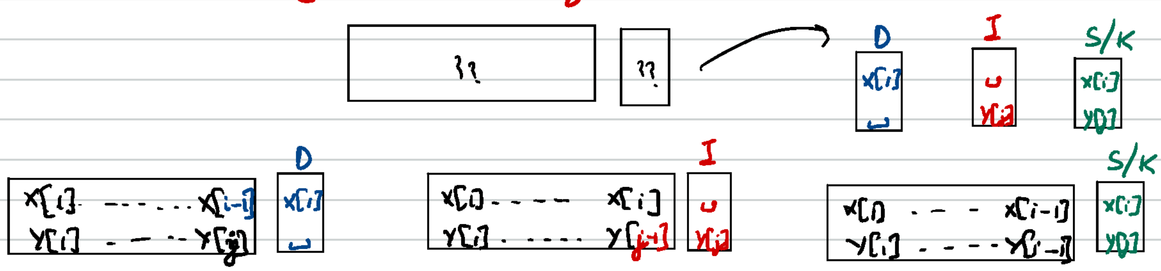
$E(i,j) = E(x[1:i], y[1:j])$ = Edit distance between $x[1...i]$ & $y[1...j]$

Example: $E(\phi, S)$, $E(SUN, SNO)$, $E(SUNNY, SNOWY)$

ϕ → empty string

Step 2: Recurrence relation

edit $x[1...i] \longrightarrow y[1...j]$

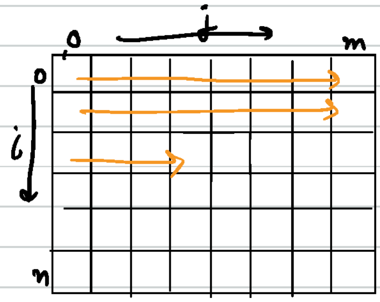


$$E(i,j) \xrightarrow{\min} \begin{cases} 1 + E(i-1, j) \\ 1 + E(i, j-1) \\ \text{diff}(x[i], y[j]) + E(i-1, j-1) \end{cases}$$

$$\text{diff}(a,b) \rightarrow \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases}$$

Base Case $E(0,0) = 0$ $\forall j$ $E(0,j) = j = E(j,0)$

Step 3: Pick the order of computation



Dependency

$E(i-1,j-1)$ $E(i-1,j)$

$E(i,j-1) \rightarrow \boxed{} E(i,j)$

	ϕ	S	N	O	W	Y
ϕ	0	1	2	3	4	5
S	1	0	1	2	3	4
O	2	1				
N	3	2				
N	4	3				
Y	5	4				

: Algorithm :

Init: For all $i = 1 \dots n$ $\quad E(i,0) = i$
For all $j = 1 \dots m$ $\quad E(0,j) = j$

For all $i = 1 \dots n$
For all $j = 1 \dots m$

$$E(i,j) = \min \begin{cases} 1 + E(i-1,j) \\ 1 + E(i,j-1) \\ diff(x_i,y_j) + E(i-1,j-1) \end{cases}$$

Run-time : $O(nm)$

Knapsack Problem

Input: Total weight capacity of a knapsack W
List of n items with weights $w_1 \quad w_2 \quad \dots \quad w_n$ (integers)
values $v_1 \quad v_2 \quad \quad v_n$ (integers)

Goal: Find set of items that will maximize total value
with total weight $\leq W$

Example:

Item	Weight	Value
1	6	\$30
2	3	\$14
3	4	\$16
4	2	\$9

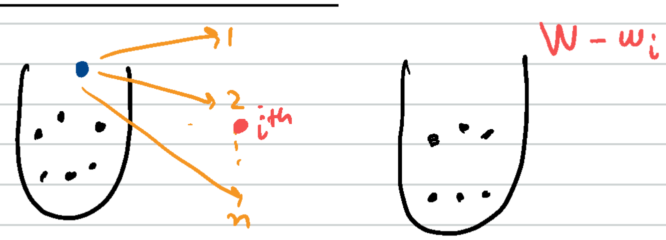
$W = 10$

Two variations: with repetition & without repetition.

Without replacement : 1 & 3 → \$46

With replacement : 1 & 4 & 4 → \$48

Knapsack with Replacement



Step1: Subproblem

$K(C)$ = max value that can be packed in a bag of capacity C.

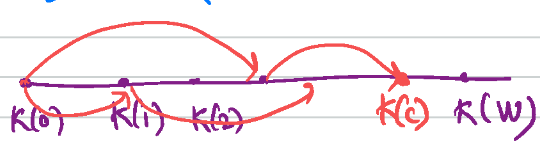
$C = 0 \dots W$

Step2:

$$K(C) = \max_{i:\, C \geq w_i} \{ v_i + K(C - w_i) \}$$

Base Case: $K(0) = 0$

Step3:

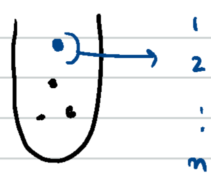


Algorithm

Input: W, $v[1 \ldots n]$, $w[1 \ldots n]$
$K(0) = 0$ For $C = 1$ to W $K(C) = \max_{i : w_i \le C} \{ v_i + K(C - w_i) \}$ Output $K(W)$

Runtime : $O(n \underbrace{W}_{})$ → exponential in $\log W$

Knapsack without repetition



Step 1: $K(c, j)$: Maximum value that can be carried in a bag of capacity c using items $1 \ldots j$.

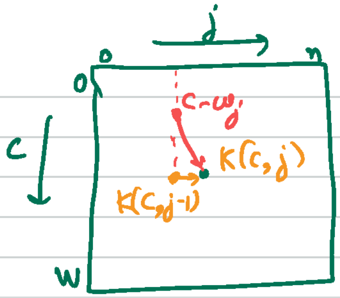
$c \to 0 \ldots W$, $j \to 1 \ldots n$

Step 2:

$$K(c, j) \to \begin{cases} K(c, j-1) & \text{if } c < w_j \\ \max\{\underbrace{K(c, j-1)}_{\text{j^{th} item is not part of the optimal solution.}}, \underbrace{v_j + K(c - w_j, j-1)}_{\text{placing j^{th} item}}\} & \text{if } c \ge w_j \end{cases}$$

Base : $K(0, j) = 0 \quad \forall j$

Step 3:



Run-time: $O(nW)$

Algorithm

Input: W, $v[1 \dots n]$, $w[1 \dots n]$

For $c = 0 \dots W$
  $K(c, 0) = 0$
For $\ell = 1 \dots n$
  For $c = 1 \dots W$
    $K(c, \ell) = \max \{ K(c, \ell-1), \underbrace{v_\ell + K(c - w_\ell, \ell-1)}_{\text{if } c - w_\ell \geq 0} \}$

Output $K(W, n)$

Chain Matrix Multiplication

Multiply $\overset{m_0\times m_1}{A} \quad \overset{m_1\times m_2}{B}$

$$(A\times B)_{ij} = \sum_{j=1}^{m_1} A_{ij}\, B_{jk}$$

$m_0\, m_1\, m_2$ multiplications.

How to multiply? $\underset{50\times 20}{A} \times \underset{20\times 1}{B} \times \underset{1\times 10}{C} \times \underset{10\times 100}{D}$

Goal: Find the best paranthezation?

$$\overbrace{(A \times \underbrace{(B\times C)}_{20\times 10})}^{50\times 10} \times D$$

$= 20\times 1\times 10 + 50\times 20\times 10 + 50\times 10\times 100$

$= 60{,}200$

$$A\times\overbrace{(\underbrace{(B\times C)}_{20\times 10}\times D)}^{20\times 100}$$

$= 200 + 20\times 10\times 100 + 50\times 20\times 100$

$= 120{,}200$

$$(A\times B)\times(C\times D)$$

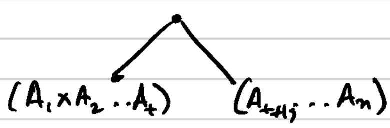
$\vdots$

$= 7000$

Input: $\overset{m_0\times m_1}{A_1} \quad \overset{m_1\times m_2}{A_2} \ \ldots \ \overset{m_{n-1}\times m_n}{A_n}$

Output: Minimum number of multiplications needed.

Step 1:



$$M(1\ldots n) = \max_t \{ M(1\ldots t) + M(t{+}1 \ldots n) + m_0\, m_t\, m_n \}$$

$A_i \ldots A_j$

Subproblem: $M(i,j)$ = the minimum number of multiplications needed to multiply $A_i \times A_{i+1} \ldots A_j$

$M(1, n) \rightarrow$ what we want!

Step 2:

$$M(i,j) = \min_{i\le k\le j} \{ M(i,k) + M(k{+}1, j) + m_{i-1}\, m_k\, m_j \}$$

$M(i,i) = 0$ for all $i \in \{1..n\}$

$A_i \times (A_{i+1} \ldots A_j)$

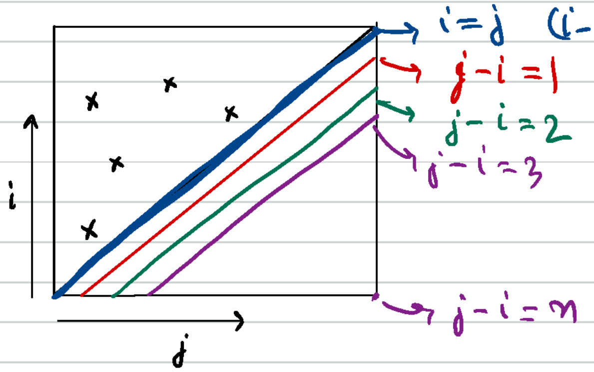
$(A_i \times A_{i+1}) \times (A_{i+2} \ldots A_j)$

$(A_i \ldots A_{j-1}) \times A_j$

Step3: Order of computation

$M(i,j)$
$j \geq i$

$M(i,j)$ $i,j < k$



$M(1,n)$

$\square \to \underline{i,j} \Rightarrow i-j=k$

Runtime $\to O(n^2 \times n) = O(n^3)$

Algorithm

For $i=1, \ldots n$ $M(i,i)=0$
For $s=1, \ldots n-1$
For $i=1 \ldots n-s$
$j=i+s$

$$M(i,j) = \min_{k=i..j} \{M(i,k) + M(k+1,j) + m_{i-1}\, m_k\, m_j\}$$

Return $M(1,n)$

Running time: $O(n^2 \times n)$